UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/855,199	05/14/2001	Vijaya Raghavan	04899-044001	8175

74321 7590 08/05/2008

LAHIVE & COCKFIELD, LLP/THE MATHWORKS
FLOOR 30, SUITE 3000
One Post Office Square
Boston, MA 02109-2127

EXAMINER	
ALHIJA, SAIF A	

ART UNIT	PAPER NUMBER
2128	

MAIL DATE	DELIVERY MODE
08/05/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	09/855,199	RAGHAVAN ET AL.	
	Examiner	Art Unit	
	SAIF A. ALHIJA	2128	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *4/30/08*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-2, 4-6, 12-13, 15-18, 21-24, 28-30, 32, 34-40, 42-49, 51-58, and 60-78* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-2, 4-6, 12-13, 15-18, 21-24, 28-30, 32, 34-40, 42-49, 51-58, and 60-78* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *24 August 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-2, 4-6, 12-13, 15-18, 21-24, 28-30, 32, 34-40, 42-49, 51-58, and 60-78 have been presented for examination.

Claims 3, 7-11, 14, 19, 20, 25-27, 31, 33, 41, 50, and 59 have been cancelled.

Response to Arguments

2. Applicant's arguments with respect to claims 1-6, 12-18, and 20-78 have been considered but are moot in view of the new ground(s) of rejection.

i) Examiner thanks the Applicant for agreeing to the change of title. An amendment to that effect is still required.

ii) Following Applicants amendments a 112 2nd rejection has been provided below.

iii) Examiner has cited particular columns and line numbers in the references applied to the claims for the convenience of the applicant. Although the specified citations are representative of the teachings of the art and are applied to specific limitations within the individual claim, other passages and figures may apply as well. It is respectfully requested from the applicant in preparing responses, to fully consider the references in their entirety as potentially teaching all or part of the claimed invention, as well as the context of the passage as taught by the prior art or disclosed by the Examiner.

iv) The Examiner respectfully requests, in the event the Applicants choose to amend or add new claims, that such claims and their limitations be directly mapped to the specification, which provides support for the subject matter. This will assist in expediting compact prosecution.

v) Further, the Examiner respectfully encourages Applicants to direct the specificity of their response with regards to this office action to the broadest reasonable interpretation of the claims as presented. This will avoid issues that would delay prosecution such as limitations not explicitly presented in the claims, intended use statements that carry no patentable weight, mere allegations of patentability, and novelty that is not clearly expressed.

Claim Rejections - 35 USC § 112

The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

3. Claims 12-16 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 12 recites the limitation "at least one state and at least one transition." There is insufficient antecedent basis for this limitation in the claim since the claim previously recites at least one state **or** transition.

Appropriate correction is required.

All claims dependent upon a rejected base claim are rejected by virtue of their dependency.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

The factual inquiries set forth in *Graham* **v.** *John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

4. Claim(s) 1-2, 4-6, 12-13, 15-18, 21-24, 28-30, 32, 34-40, 42-49, 51-58, and 60-78 is rejected under 35 U.S.C. 103(a) as being unpatentable over **Kodosky et al. "System and Method for Programmatically Generating**

a Graphical Program in Response to a State Diagram" U.S. Patent Application Publication 2002/0083413, hereafter Kodosky in view of **Stateflow Version 3.0 (R11), hereafter Stateflow.**

Regarding Claim 1:

Kodosky et al. discloses a computer-implemented method comprising:

providing a graphical user interface for defining a function to be used in a graphical representation of a finite state machine where the graphical representation is an executable model of the finite state machine and includes at least one state and at least one transition; **(Page 14, Paragraph 165, Lines 1-5. Figure 19)**

representing the function graphically such that the function is graphically represented separately from the at least one state and at least one transition in the graphical representation of the finite state machine, wherein the function that is represented graphically has a, and the function is defined in a graphical language; and **(Page 14, Paragraph 165, Lines 1-5. Figure 19)**

calling the function that is represented graphically by the function name according to the syntax specified by the function prototype from within the graphical representation of the finite state machine. **(Page 15, Paragraph 166, Lines 11-15)**

Kodosky does not explicitly recite a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function.

However Stateflow recites a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function. **(Stateflow. Page 2, Figure at top showing a graphical function as well as its syntax and prototype)**

Kodosky and Stateflow are analagous art in the field of graphical programming.

It would have been obvious to one of ordinary skill in the art at the time of the invention to utilize the graphical function prototyping in Stateflow for the graphical programming in Kodosky since "a graphical function is easier to create, access, and manage" in a graphical environment and function prototypes are a well known method of function modularity as well as interface/implementation separation. (Stateflow. "Why Use a Graphical Function?")

Regarding Claim 2:

Kodosky et al. discloses the method of claim 1 wherein the graphical function further includes a function block. **(Page 14, Paragraph 165, Lines 7-9. Figure 19)**

Regarding Claim 4:

Kodosky et al. discloses the method of claim 1 wherein the graphical function further includes a function flow diagram that graphically defines a procedure performed by the function. **(Page 1, Paragraph 9, Lines 7-9)**

Regarding Claim 5: a computer-implemented method, said method comprising

providing a graphical user interface for defining a function to be used in a graphical representation of a finite state machine where the graphical representation is an executable model of the finite state machine and includes at least one state and at least one transition **(Page 14, Paragraph 165, Lines 1-5. Figure 19)**

representing the function graphically such that the function is graphically represented separately from the at least one state and at least one transition in the graphical representation of the finite state machine, wherein the function is represented graphically as a diagram comprising graphical elements and has a function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function; and **(Page 14, Paragraph 165, Lines 1-5. Figure 19)**

calling the function that is represented graphically by the function name according to the syntax specified by the function prototype from within the finite state machine. **(Page 15, Paragraph 166, Lines 11-15)**

Kodosky does not explicitly recite a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function.

However Stateflow recites a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function. **(Stateflow. Page 2, Figure at top showing a graphical function as well as its syntax and prototype)**

Kodosky and Stateflow are analagous art in the field of graphical programming.

It would have been obvious to one of ordinary skill in the art at the time of the invention to utilize the graphical function prototyping in Stateflow for the graphical programming in Kodosky since "a graphical

function is easier to create, access, and manage" in a graphical environment and function prototypes are a well known method of function modularity as well as interface/implementation separation. (Stateflow. "Why Use a Graphical Function?")

Regarding Claim 6:

Kodosky et al. discloses the method of claim 1 further comprising modifying the function through graphical diagramming. **(Figure 8)**

Regarding Claim 12:

Kodosky et al. discloses a computer program product, stored in a computer readable storage medium, comprising instructions to cause a computer to:

receive input defining a graphical function for use in a finite state machine, the graphical function having a function prototype that specifies a syntax for invoking the graphical function, the function prototype specifying a function name for the graphical function; **(Page 14, Paragraph 165, Lines 1-5. Figure 19)**

use the graphical function in a simulation of a system represented by the finite state machine, wherein the instructions to use the graphical function further comprise instructions to call the graphical function by the function name according to the syntax specified by the function prototype from at least one state or transition in the finite state machine. **(Page 14, Paragraph 165, Lines 1-5. Figure 19)**

the graphical function being represented graphically such that the graphical function is graphically represented separately from the at least one state and at least one transition in the finite state machine. **(Page 14, Paragraph 165, Lines 1-5. Figure 19)**

Kodosky does not explicitly recite a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function.

However Stateflow recites a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function. **(Stateflow. Page 2, Figure at top showing a graphical function as well as its syntax and prototype)**

Kodosky and Stateflow are analagous art in the field of graphical programming.

It would have been obvious to one of ordinary skill in the art at the time of the invention to utilize the graphical function prototyping in Stateflow for the graphical programming in Kodosky since "a graphical function is easier to create, access, and manage" in a graphical environment and function prototypes are a well known method of function modularity as well as interface/implementation separation. (Stateflow. "Why Use a Graphical Function?")

Regarding Claim 13:

Kodosky et al. discloses the computer program product of claim 12 wherein the input defining the graphical function is entered into a function block**. (Page 1, Paragraph 9, Lines 1-2)**

Regarding Claim 15:

Kodosky et al. discloses the computer program product of claim 12 wherein the input comprises a function flow diagram that graphically defines a procedure performed by the function. **(Page 1, Paragraph 9, Lines 7-9)**

Regarding Claim 16:

Kodosky et al. discloses the computer program product of claim 15 wherein the function flow diagram is a comprised of graphical elements. **(Figure 8)**

Regarding Claim 17:

Kodosky et al. discloses a system for modeling at least one finite state machine said system comprising:

a computer comprising a graphical user interface, a memory, a storage, and at least one input device; **(Page 6, Paragraph 63, Lines 1-4)**

means to receive input to define a graphical function, the graphical function having a function prototype that specifies a syntax for invoking the graphical function, the function prototype specifying a function name for the graphical function; **(Page 6, Paragraph 63, Lines 1-8)**

means to represent the graphical function as an executable state flow diagram; **(Page 2, Paragraph 16, Lines 4-9)**

means to call the graphical function by the function name according to the syntax specified by the function prototype from the at least one finite state machine in a simulation of the at least one finite state machine the at least one finite state machine including at least one state and at least one transition the graphical function being represented graphically such that the graphical function is graphically represented separately from the at least one state and at least one transition in the finite state machine. **(Page 15, Paragraph 166, Lines 13-20)**

Kodosky does not explicitly recite a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function.

However Stateflow recites a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function. **(Stateflow. Page 2, Figure at top showing a graphical function as well as its syntax and prototype)**

Kodosky and Stateflow are analagous art in the field of graphical programming.

It would have been obvious to one of ordinary skill in the art at the time of the invention to utilize the graphical function prototyping in Stateflow for the graphical programming in Kodosky since "a graphical function is easier to create, access, and manage" in a graphical environment and function prototypes are a well known method of function modularity as well as interface/implementation separation. (Stateflow. "Why Use a Graphical Function?")

Regarding Claim 18:

Kodosky et al. discloses the system of claim 17 wherein the input to define the graphical function is entered into a function block. **(Page 1, Paragraph 9, Lines 1-2)**

Regarding Claim 21:

Kodosky et al. discloses the system of claim 17 wherein the input comprises a function flow diagram that graphically defines a procedure performed by the function. **(Page 1, Paragraph 9, Lines 7-9)**

Regarding Claim 22:

Kodosky et al. discloses the system of claim 21 wherein the function flow diagram is comprised of graphical elements. **(Figure 8)**

Regarding Claim 23:

Kodosky et al. discloses the system of claim 21 further comprising means for hiding the display of the function flow diagram based upon input. **(Page 1, Paragraph 9, Lines 7-9)**

Regarding Claim 24:

Kodosky et al. discloses a method of operating a data processing system having a graphical user interface said method comprising:

creating a graphical representation of a finite state machine and a graphical representation of a function for use in the graphical representation of the finite state machine , the function having a function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function, the graphical representation of the finite state machine including at least one state and at least one transition the graphical representation of the function being represented graphically such that the function is graphically represented separately from the at least one state and at least one transition in the graphical representation of the finite state machine, **(Page 1, Paragraph 9, Lines 9-14)**

simulating a system represented by the finite state machine wherein the graphical representation of the finite state machine is an executable model of the system; and **(Page 1, Paragraph 10, Lines 10-13)**

calling the function by the function name according to the syntax specified by the function prototype from the executable model of the system during the act of simulating the system represented by the finite state machine **(Page 14, Paragraph 165, Lines 1-5. Paragraphs 168-172. Figure 19)**

Kodosky does not explicitly recite a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function.

However Stateflow recites a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function. **(Stateflow. Page 2, Figure at top showing a graphical function as well as its syntax and prototype)**

Kodosky and Stateflow are analagous art in the field of graphical programming.

It would have been obvious to one of ordinary skill in the art at the time of the invention to utilize the graphical function prototyping in Stateflow for the graphical programming in Kodosky since "a graphical function is easier to create, access, and manage" in a graphical environment and function prototypes are a well known method of function modularity as well as interface/implementation separation. (Stateflow. "Why Use a Graphical Function?")

Regarding Claim 28:

Kodosky et al. discloses the method of claim 24 further comprising shadowing the function, wherein shadowing comprising using in a function invocation a function definition closest to a point of invocation of the function in a state diagram hierarchy. **(Page 3, Paragraph 20, Lines 8-13; Creators priority order can allow for closest function definition to execute.)**

Regarding Claim 29

Kodosky et al. discloses the method of claim 24 wherein the function is exportable by a state chart and may be invoked anywhere in the finite state machine in which the chart appears, including other charts that define the finite state machine. **(Page 3, Paragraph 26, Lines 4-10. Page 9, Paragraph 100, Lines 1-5)**

Regarding Claim 30:

Kodosky et al. discloses the method of claim 24 wherein simulating the system represented by the finite state machine further comprises computer code generation. **(Page 12, Paragraph 133, Lines 1-4)**

Regarding Claim 32:

Kodosky et al. discloses a computer readable storage medium having encoded thereon

instructions for causing a computer system to receive through a graphical user interface graphical representation of a finite state machine and a graphical representation of a function for use in the graphical representation of the finite state machine, the function having a function prototype that specifies a syntax for

invoking the function, the function prototype specifying a function name for the function the graphical representation of the finite state machine including at least one state and at least one transition, the function being represented graphically such that the function is graphically represented separately from the at least one state and the at least one transition in the graphical representation of the finite state machine; and **(Page 1, Paragraph 9, Lines 9-14)**

instructions for simulating a system represented by the finite state machine where the graphical representation is an executable model of the system; and **(Page 1, Paragraph 10, Lines 10-13)**

instructions for calling the function by the function name according to the syntax specified in the function prototype from at least one place in the executable model during the system simulation. **(Page 14, Paragraph 165, Lines 1-5. Paragraphs 168-172. Figure 19)**

Kodosky does not explicitly recite a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function.

However Stateflow recites a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function. **(Stateflow. Page 2, Figure at top showing a graphical function as well as its syntax and prototype)**

Kodosky and Stateflow are analagous art in the field of graphical programming.

It would have been obvious to one of ordinary skill in the art at the time of the invention to utilize the graphical function prototyping in Stateflow for the graphical programming in Kodosky since "a graphical function is easier to create, access, and manage" in a graphical environment and function prototypes are a well known method of function modularity as well as interface/implementation separation. (Stateflow. "Why Use a Graphical Function?")

Regarding Claim 34:

Kodosky et al. discloses in an electronic device, a method of graphically representing an event-driven system, said method comprising:

Providing one or more block components representing one or more states in an executable model; **(Page 1, Paragraph 9, Lines 1-3)**

Providing one or more transition components representing transitions between the one or more block states; **(Page 2, Paragraph 16, Lines 1-4)** and

Providing a function, having a function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function said function comprising at least two graphical components and being referenced by at least one the states or at least one of the transitions to call the function by the function name according to the syntax specified by the function prototype at the at least one of the states or the at least one of the transitions the function being represented graphically such that the function is graphically represented separately from the at least one of the state and the at least one of the transition in the executable model. **(Page 2, Paragraph 17, Lines 1-4)**

Kodosky does not explicitly recite a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function.

However Stateflow recites a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function. **(Stateflow. Page 2, Figure at top showing a graphical function as well as its syntax and prototype)**

Kodosky and Stateflow are analagous art in the field of graphical programming.

It would have been obvious to one of ordinary skill in the art at the time of the invention to utilize the graphical function prototyping in Stateflow for the graphical programming in Kodosky since "a graphical function is easier to create, access, and manage" in a graphical environment and function prototypes are a well known method of function modularity as well as interface/implementation separation. (Stateflow. "Why Use a Graphical Function?")

Regarding Claim 35:

Kodosky et al. discloses the method of claim 34, wherein the function accepts at least one argument and returns at least one result. **(Page 1, Paragraph 9, Lines 1-4)**

Regarding Claim 36:

Kodosky et al. discloses the method of claim 34, further comprising invoking the function at a second one of the one or more transition components or one or more block components. **(Page 12, Paragraph 132 Lines 1-5. Figure 8. Page 1, Paragraph 10, Lines 1-5)**

Regarding Claim 37:

Kodosky et al. discloses the method of claim 34 further comprising specifying data properties of the function. **(Page 1, Paragraph 9, Lines 7-9)**

Regarding Claim 38:

Kodosky et al. discloses the method of claim 34 further comprising associating a data item with the function. **(Page 1, Paragraph 9, Lines 7-9. Page 2, Paragraph 11, Lines 2-7)**

Regarding Claim 39:

Kodosky et al. discloses the method of claim 34, wherein the function comprises a graphical function. **(Page 6, Paragraph 63, Lines 1-8)**

Regarding Claim 40:

Kodosky et al. discloses the method of claim 34, wherein the function has a plurality of configurable properties. **(Page 1, Paragraph 10, Lines 1-5)**

Regarding Claim 42:

Kodosky et al. discloses the method of claim 34, further comprising providing a shadowing function, wherein shadowing comprises using in a function invocation a function definition proximally closest to a point of invocation of the function in a state diagram hierarchy. **(Page 3, Paragraph 20, Lines 8-13; Creators priority order can allow for closest function definition to execute.)**

Regarding Claim 43:

Kodosky et al. discloses in a graphical representation environment, a system for graphically representing an event-driven system, said system comprising:

One or more bock components representing one or more states in an executable model; **(Page 1, Paragraph 9, Lines 1-3)**

One or more transition components representing transitions between the one or more block components representing the one or more states; **(Page 2, Paragraph 16, Lines 1-4) and**

A component representing a graphical function having a function prototype that specifies a syntax for invoking the function the function prototype specifying a function name for the graphical function and referenced by at least one of the states or at least one of the transitions to call the graphical function by the function name according to the syntax specified by the function prototype at one of the states or one of the transition the graphical function being represented graphically such that the graphical function is graphically represented separately from the at least one of the states and the at least one of the transitions in the executable model. **(Page 2, Paragraph 17, Lines 1-4)**

Kodosky does not explicitly recite a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function.

However Stateflow recites a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function. **(Stateflow. Page 2, Figure at top showing a graphical function as well as its syntax and prototype)**

Kodosky and Stateflow are analagous art in the field of graphical programming.

It would have been obvious to one of ordinary skill in the art at the time of the invention to utilize the graphical function prototyping in Stateflow for the graphical programming in Kodosky since "a graphical function is easier to create, access, and manage" in a graphical environment and function prototypes are a well known method of function modularity as well as interface/implementation separation. (Stateflow. "Why Use a Graphical Function?")

Regarding Claim 44:

Kodosky et al. discloses the system of claim 43, wherein the graphical function accepts at least one argument and returns at least one result. **(Page 1, Paragraph 9, Lines 1-4)**

Regarding Claim 45:

Kodosky et al. discloses the system of claim 43, wherein at least a subset of the one or more block components representing the states and the one or more transition components can invoke the graphical function. **(Page 12, Paragraph 132 Lines 1-5. Figure 8. Page 1, Paragraph 10, Lines 1-5)**

Regarding Claim 46:

Kodosky et al. discloses the system of claim 43, further comprising means for specifying data properties of the graphical function. **(Page 1, Paragraph 9, Lines 7-9)**

Regarding Claim 47:

Kodosky et al. discloses the system of claim 43, further comprising means for associating a data item with the graphical function. **(Page 1, Paragraph 9, Lines 7-9. Page 2, Paragraph 11, Lines 2-7)**

Regarding Claim 48:

Kodosky et al. discloses the system of claim 43, wherein the component representing the function is referenced by one more of: at least one of the states or at least on of the transitions. **(Page 6, Paragraph 63, Lines 1-8)**

Regarding Claim 49:

Kodosky et al. discloses the system of claim 43, wherein the graphical function has a plurality of configurable properties. **(Page 1, Paragraph 10, Lines 1-5)**

Regarding Claim 51:

Kodosky et al. discloses the system of claim 43, further comprising means for providing a shadowing function, wherein shadowing comprises using in a function invocation a function definition proximally closest to a point of invocation of the function in a state diagram hierarchy. **(Page 3, Paragraph 20, Lines 8-13; Creators priority order can allow for closest function definition to execute.)**

Regarding Claim 52:

Kodosky et al. discloses a computer readable storage medium for use in a graphical representation environment on an electronic device, the medium holding instructions executable using the electronic device for graphically representing an event-driven system, said instructions comprising instructions of:

Providing one or more block components representing one or more states in an executable model; **(Page 1, Paragraph 9, Lines 1-3)**

Providing one or more transition components representing transitions between the one or more block components representing the states; **(Page 2, Paragraph 16, Lines 1-4)** and

Providing a block component representing a graphical function having a function prototype that specifies a syntax for invoking the graphical function, the function prototype specifying a function name for the graphical function and reference by at least one of the states or at least one of the transitions to call the graphical function by the function name according to the syntax specified by the function prototype at one of the states or one of the transitions during execution of the event-driven system, the graphical function being represented graphically such that the graphical function is graphically represented separately from the at least one of the state and the at least one of the transitions in the executable model. **(Page 2, Paragraph 17, Lines 1-4)**

Kodosky does not explicitly recite a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function.

However Stateflow recites a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function. **(Stateflow. Page 2, Figure at top showing a graphical function as well as its syntax and prototype)**

Kodosky and Stateflow are analagous art in the field of graphical programming.

It would have been obvious to one of ordinary skill in the art at the time of the invention to utilize the graphical function prototyping in Stateflow for the graphical programming in Kodosky since "a graphical function is easier to create, access, and manage" in a graphical environment and function prototypes are a well known method of function modularity as well as interface/implementation separation. (Stateflow. "Why Use a Graphical Function?")

Regarding Claim 53:

Kodosky et al. discloses the computer readable storage medium of claim 52, wherein the graphical function accepts at least one argument and returns at least one result. **(Page 1, Paragraph 9, Lines 1-4)**

Regarding Claim 54:

Kodosky et al. discloses the computer readable storage medium of claim 52, wherein the one or more transition components can invoke the graphical function. **(Paragraph 132 Lines 1-5. Figure 8. Page 1, Paragraph 10, Lines 1-5)**

Regarding Claim 55:

Kodosky et al. discloses the computer readable storage medium of claim 52, further comprising instructions for accepting input specifying data properties of the graphical function. **(Page 1, Paragraph 9, Lines 7-9)**

Regarding Claim 56:

Kodosky et al. discloses the computer readable storage medium of claim 52, further comprising instructions for associating a data item with the graphical function. **(Page 1, Paragraph 9, Lines 7-9. Page 2, Paragraph 11, Lines 2-7)**

Regarding Claim 57:

Kodosky et al. discloses the computer readable storage medium of claim 52, wherein the graphical function comprises two or more graphical elements. **(Page 6, Paragraph 63, Lines 1-8)**

Regarding Claim 58:

Kodosky et al. discloses the computer readable storage medium of claim 52, wherein the graphical function has a plurality of configurable properties. **(Page 1, Paragraph 10, Lines 1-5)**

Regarding Claim 60:

Kodosky et al. discloses the computer readable storage medium of claim 52, further comprising instructions for providing a shadowing function wherein shadowing comprises using in a function invocation a function definition proximally closest to a point of invocation of the graphical function in a state diagram hierarchy. **(Page 3, Paragraph 20, Lines 8-13; Creators priority order can allow for closest function definition to execute.)**

Regarding Claim 61:

Kodosky et al. discloses A computer implemented method for modeling a system using a graphical block diagram environment, said method comprising:

graphically representing a function having a function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function for use in an executable model within the graphical block diagram environment the executable model including at least one state and at least one transition, the function being represented graphically such that the function is graphically represented separately from the at least one state and the at least one transition in the executable model; and; **(Page 14, Paragraph 165, Lines 1-5. Figure 19)** and

textually referencing the graphically represented function by the function name according to the syntax specified by the function prototype within the model to cause an invocation of the graphically represented function during execution of the model. **(Paragraph 132 Lines 1-5. Figure 8)**

Kodosky does not explicitly recite a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function.

However Stateflow recites a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function. **(Stateflow. Page 2, Figure at top showing a graphical function as well as its syntax and prototype)**

Kodosky and Stateflow are analagous art in the field of graphical programming.

It would have been obvious to one of ordinary skill in the art at the time of the invention to utilize the graphical function prototyping in Stateflow for the graphical programming in Kodosky since "a graphical function is easier to create, access, and manage" in a graphical environment and function prototypes are a well known method of function modularity as well as interface/implementation separation. (Stateflow. "Why Use a Graphical Function?")

Regarding Claim 62:

Kodosky et al. discloses The computer implemented method of claim 61, wherein the model is represented as a finite state machine. **(Page 3, Paragraph 20, Lines 8-13)**

Regarding Claim 63:

Kodosky et al. discloses The computer implemented method of claim 62, wherein the finite state machine is a hierarchical finite state machine. **(Page 3, Paragraph 20, Lines 8-13)**

Regarding Claim 64:

Kodosky et al. discloses The computer implemented method of claim 62 further comprising:

Associating the graphically represented function with at least one state or transition within the finite state machine. **(Page 2, Paragraph 16, Lines 1-4)**

Regarding Claim 65:

Kodosky et al. discloses The computer implemented method of claim 61, wherein the graphically represented function is represented as at least one of a finite state machine, a state flow diagram, a function flow diagram, and a graphical block diagram model. **(Page 3, Paragraph 20, Lines 8-13)**

Regarding Claim 66:

Kodosky et al. discloses A computer readable storage medium holding instructions executable using the electronic device for modeling a system using a graphical block diagram environment, said instructions comprising instructions for:

Graphically defining a function having a function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function for use in an executable model within the graphical block diagram environment the executable model including at least one state or transition, the function being represented graphically such that the function is graphically represented separately from the at least one state and at least one transition in the executable model; and **(Page 14, Paragraph 165, Lines 1-5. Figure 19)**

textually referencing the graphically represented function by the function name according to the syntax specified by the function prototype within the model to cause an invocation of the graphically represented function during execution of the model. **(Paragraph 132 Lines 1-5. Figure 8)**

Kodosky does not explicitly recite a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function.

However Stateflow recites a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function. **(Stateflow. Page 2, Figure at top showing a graphical function as well as its syntax and prototype)**

Kodosky and Stateflow are analagous art in the field of graphical programming.

It would have been obvious to one of ordinary skill in the art at the time of the invention to utilize the graphical function prototyping in Stateflow for the graphical programming in Kodosky since "a graphical function is easier to create, access, and manage" in a graphical environment and function prototypes are a well known method of function modularity as well as interface/implementation separation. (Stateflow. "Why Use a Graphical Function?")

Regarding Claim 67:

Kodosky et al. discloses The computer readable storage medium of claim 66, wherein the model is represented as a finite state machine. **(Page 3, Paragraph 20, Lines 8-13)**

Regarding Claim 68:

Kodosky et al. discloses The **computer readable storage medium** of claim 67 further comprising instructions for:

Associating the graphically represented function with at least one state of transition within the finite state machine. **(Page 2, Paragraph 16, Lines 1-4)**

Regarding Claim 69:

Kodosky et al. discloses The computer readable storage medium of claim 66, wherein the graphically represented function is represented as at least one or a combination of:

a finite state machine, **(Page 3, Paragraph 20, Lines 8-13)**

a state flow diagram,

a function flow diagram,

and a graphical block diagram model.

Regarding Claim 70:

Kodosky et al. discloses A computer implemented system for modeling using a graphical block diagram environment, said system comprising:

Means for representing a function having a function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function, the function being defined graphically for use in an executable model within the graphical block diagram environment the executable model including at least one state and at least one transition, the function being represented graphically such that the function is graphically represented separately from the at least one state and at least one transition in the executable model; and **(Page 14, Paragraph 165, Lines 1-5. Figure 19)**

Means for textually referencing the function defined graphically by the function name according to the syntax specified by the function prototype within the model to cause an invocation of the function during execution of the model. **(Paragraph 132 Lines 1-5. Figure 8)**

Kodosky does not explicitly recite a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function.

However Stateflow recites a function prototype and the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function. **(Stateflow. Page 2, Figure at top showing a graphical function as well as its syntax and prototype)**

Kodosky and Stateflow are analagous art in the field of graphical programming.

It would have been obvious to one of ordinary skill in the art at the time of the invention to utilize the graphical function prototyping in Stateflow for the graphical programming in Kodosky since "a graphical function is easier to create, access, and manage" in a graphical environment and function prototypes are a well known method of function modularity as well as interface/implementation separation. (Stateflow. "Why Use a Graphical Function?")

Regarding Claim 71:

Kodosky et al. discloses The system of claim 70, wherein the executable model is represented as a finite state machine. **(Page 3, Paragraph 20, Lines 8-13)**

Regarding Claim 72:

Kodosky et al. discloses The system of claim 71 further comprising means for associating the graphically represented function with at least one state of transition within the finite state machine. **(Page 3, Paragraph 20, Lines 8-13)**

Regarding Claim 73:

Kodosky et al. discloses The system of claim 70, wherein the graphically represented function is represented as at least one or a combination of a finite state machine, a state flow diagram, a function flow diagram, and a graphical block diagram model. **(Page 3, Paragraph 20, Lines 8-13)**

Regarding Claim 74:

Kodosky et al. discloses A graphical block diagram modeling system comprising:

A graphical function for use in an executable model, the graphical function having a function prototype that specifies a syntax for invoking the graphical function, the function prototype specifying a function name for the graphical function wherein at least a subset of commands of the graphical function are defined through a graphical representation the executable model including at least one state and at least one transition, the graphical function being represented graphically such that the graphical function is graphically represented separately from the at least one state and at least one transition in the executable model; and; **(Page 14, Paragraph 165, Lines 1-5. Figure 19)** and

A graphical representation of the model including a textual reference of the graphical function by the function name according to the syntax specified by the function prototype within the graphical representation of the model to cause an invocation of the graphical function during an execution of the model. **(Paragraph 132 Lines 1-5. Figure 8)**

Kodosky does not explicitly recite a function prototype or the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function.

However Stateflow recites a function prototype and the function prototype that specifies a syntax for invoking the function, the function prototype specifying a function name for the function. **(Stateflow. Page 2, Figure at top showing a graphical function as well as its syntax and prototype)**

Kodosky and Stateflow are analagous art in the field of graphical programming.

It would have been obvious to one of ordinary skill in the art at the time of the invention to utilize the graphical function prototyping in Stateflow for the graphical programming in Kodosky since "a graphical function is easier to create, access, and manage" in a graphical environment and function prototypes are a

well known method of function modularity as well as interface/implementation separation. (Stateflow. "Why Use a Graphical Function?")

Regarding Claim 75:

Kodosky et al. discloses The system of claim 74, wherein the model is represented as a finite state machine. **(Page 3, Paragraph 20, Lines 8-13)**

Regarding Claim 76:

Kodosky et al. discloses The system of claim 75, wherein the finite state machine is a hierarchical finite state machine. **(Page 3, Paragraph 20, Lines 8-13)**

Regarding Claim 77:

Kodosky et al. discloses The system of claim 75, wherein the finite state machine further comprises at least one state or transition associated with the graphical function. **(Page 3, Paragraph 20, Lines 8-13)**

Regarding Claim 78:

Kodosky et al. discloses The system of claim 74, wherein the graphical function is represented as at least one or a combination of:

a finite state machine, **(Page 3, Paragraph 20, Lines 8-13)**

a state flow diagram,

a function flow diagram,

and a graphical block diagram model.

Conclusion

5. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL.** See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

6. All Claims are rejected.

7. Any inquiry concerning this communication or earlier communications from the examiner should be directed to SAIF A. ALHIJA whose telephone number is (571)272-8635. The examiner can normally be reached on M-F, 11:00-7:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kamini Shah can be reached on (571) 272-22792279. The fax phone number for the organization where this application or proceeding is assigned is (571) 273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Kamini S Shah/

Supervisory Patent Examiner, Art Unit 2128

SAA

July 23, 2008